

Anja Reusch

Dresden Database Research Group

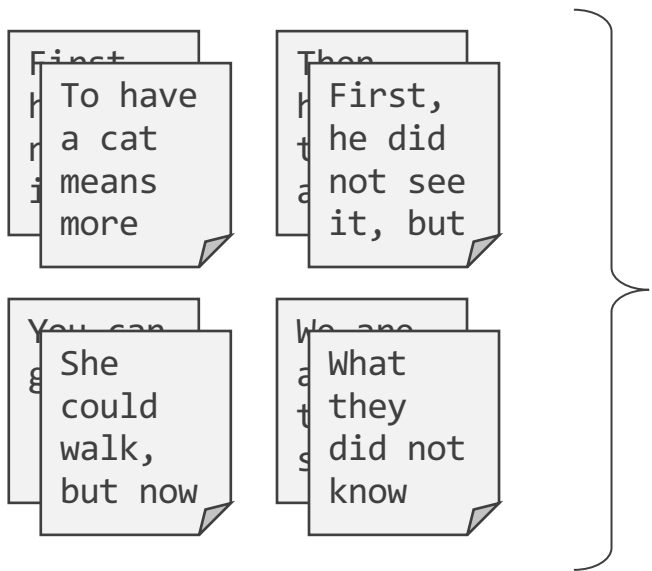
12. Text Processing

Scalable Data Engineering

Winter Semester 2022/2023

Version: 11. Januar 2023

Imagine we have ...



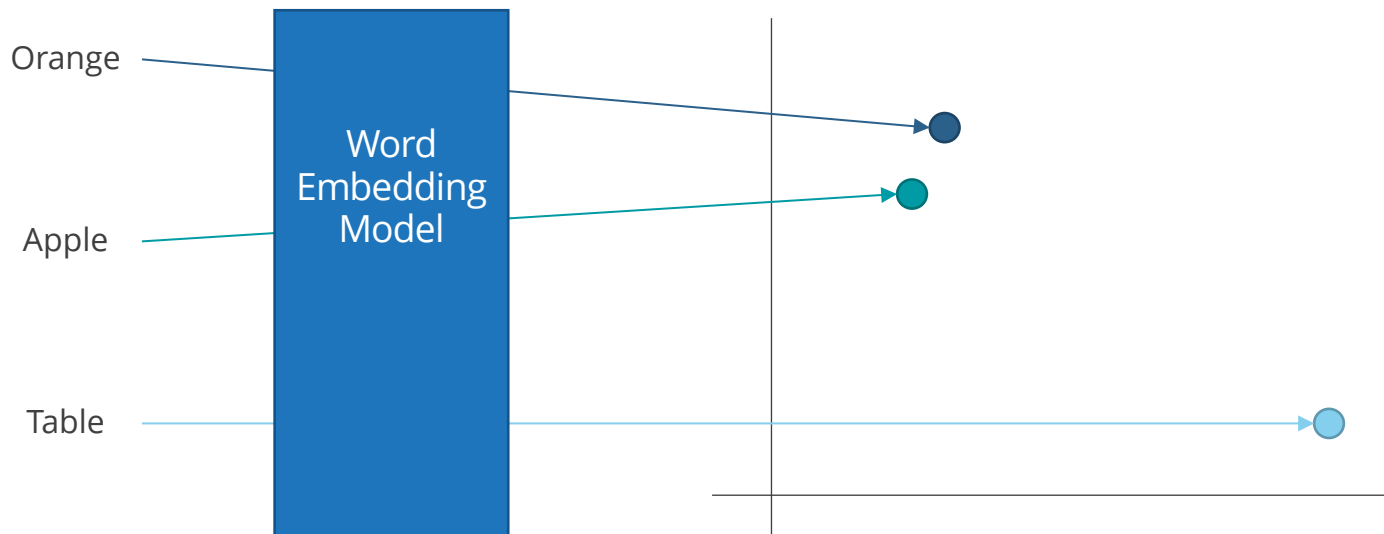
A huge amount of text: Each file is a single string

How can we find out:

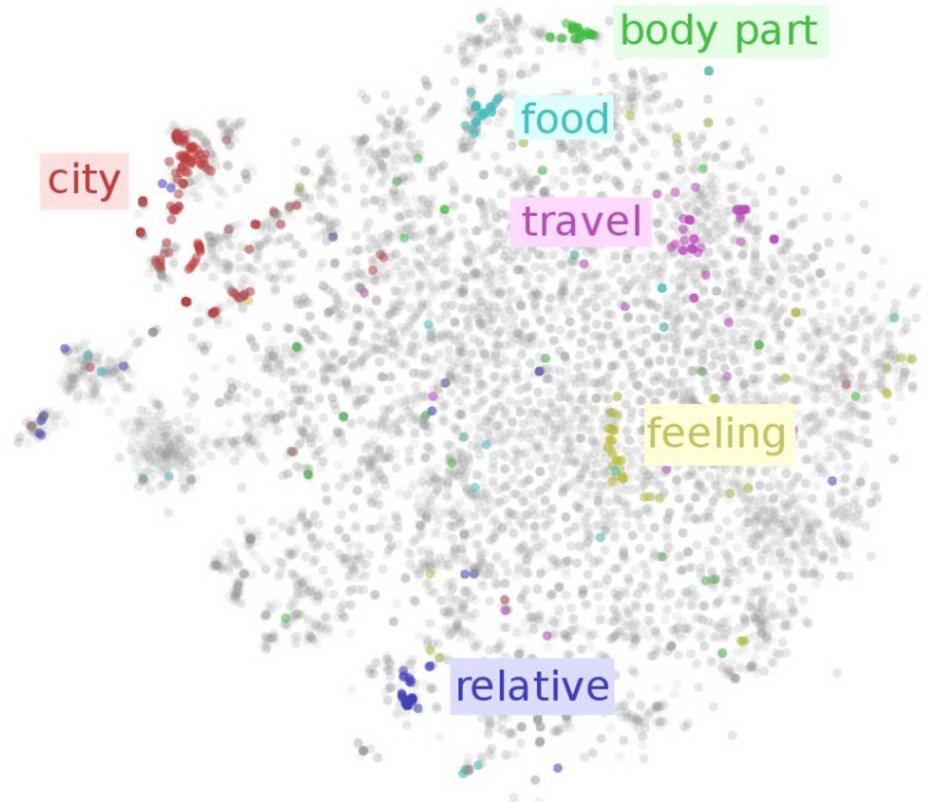
- Which files contain the word apple?
- What a file is about?
- Whether a text has a positive or negative sentiment?
- How many files talk about politics?

Goal of Embeddings

Semantic Representation of Words



Goal of Embeddings



Content

Pre-Processing

Word Embeddings

Word2Vec

Applications

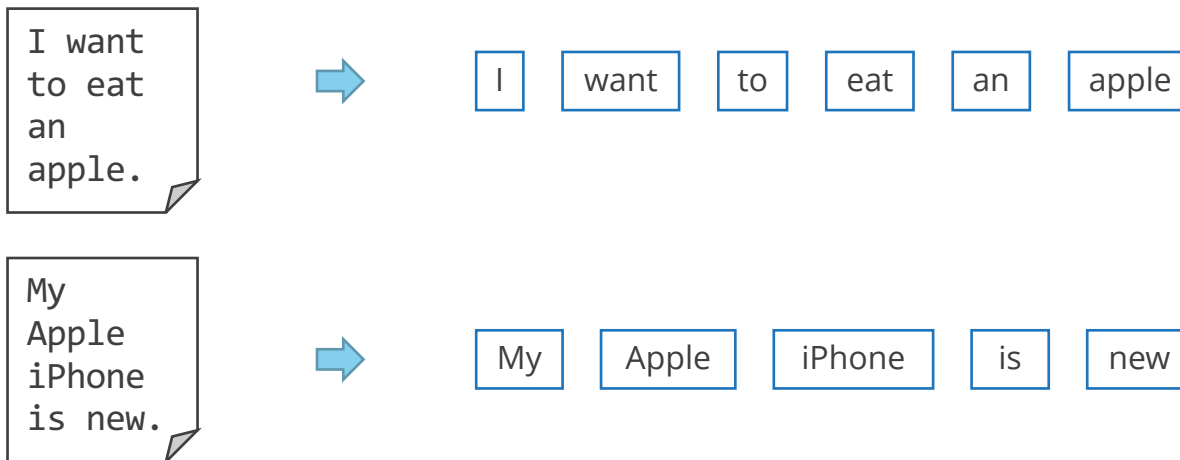
- Similarity
- Analogy



Pre-Processing

Which files contain the word apple?

What is a word?



Problems:

- Apple = apple?
- Apple iPhone: one or two tokens?
- Keep or remove punctuation?

A text is often encoded into a single string

- Tokenizing consists in **splitting this string into words** or tokens
- Needed to apply further processing, e.g. stemming and lemmatization

Notion of 'word' is not straightforward

- **Orthographic word**: string of characters with 'whitespace' at each end; e.g. "they will want to leave"
- **Phonological word**: 'words' which are pronounced as a phonological unit; e.g. "they'll wanna leave"
- **Word form**: have, has, had, having are word forms of the lemma have
- **Lemma/Citation Form**: Grammatical form that is chosen to represent a lexeme. In English, usually the base form (i.e., with no grammatical marking)
- **Multi-part/Discontinuous Words**: Sequences which are multiple orthographic words but exhibit the semantic coherence of words; e.g. Kim **rang** her **up**
- **Short Forms**: abbreviations (Dept.), logograms (£), contractions (we'll), acronyms (BBC)

Tokenization problems

One word or two or several

- Hewlett-Packard
- State-of-the-art
- Co-education
- The hold-him-back-and-drag-him-away maneuver
- O'Brian
- Data base
- San Francisco
- Los Angeles-based company
- Cheap San Francisco-Los Angeles fares
- York University vs. New York University

Tokenization problems (2)

Numbers

- 3/20/91
- 20/3/91
- Mar 20, 1991
- B-52
- 100.2.86.144
- (800) 234-2333
- 800.234.2333

Apostrophes can be part of a word, a part of a possessive or just a mistake

- Rosie O'Donnell, can't, don't, 80's, 1890's, men' straw hats, master's degree, england's ten largest cities

Tokenization problems (3)

Special characters are...

- ...an important part of tags, URLs, code in documents

Capitalized words can have different meanings from lower case words

- Bush versus bush



- Apple versus apple



Tokenization problems (4)

Chinese

- No whitespaces between words nor sentences

莎拉波娃现在居住在美国东南部的佛罗里达。今年4月9日，莎拉波娃在美国第一大城市纽约度过了18岁生日。生日派对上，莎拉波娃露出了甜美的微笑。

Tokenization problems (5)

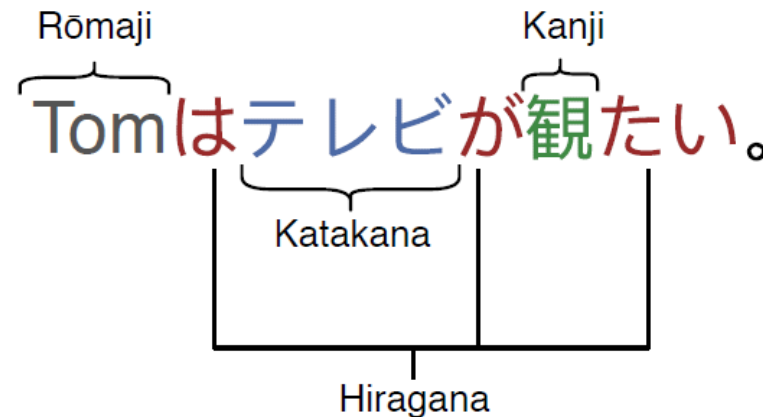
Word $\neq$ character

包		bāo	bag	bag
包	括	bāo kuò	bag + to enclose	to include
面	包	miàn bāo	face + bag	bread
手	提	shǒu tí bāo	hand + to hold + bag	handbag
打	包	dǎ bāo	to beat + bag	to pack

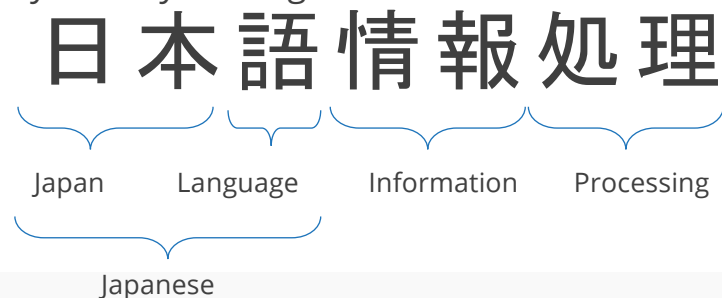
Tokenization problems (6)

Japanese

- Sentences can consist of four systems
- Chinese characters
- Hiragana syllabary for inflectional endings and functional words
- Katakana syllabary for transcript of foreign words and other uses
- Latin
- No spaces (as in Chinese)
- End user can express query entirely in hiragana!



Compound words



Tokenization problems (7)

Other cases of “no whitespace”

- Compounds in Dutch, German, Swedish
- Computerlinguistik → Computer + Linguistik
- Lebensversicherungsgesellschaftsangestellter
- → leben + versicherung + gesellschaft + angestellter
- Inuit: tusaatsiarunnangittualuujunga (I can't hear very well.)
- Many other languages with segmentation difficulties: Finnish, Urdu, . . .

Tokenization problems (9)

Accents

- résumé vs. resume (simple omission of accent)

Umlauts

- Universität vs. Universitaet (substitution with special letter sequence “ae”)

Most important criterion

- How are users likely to write their queries for these words?
- Even in languages that standardly have accents, users often do not type them (Polish?)

Note

- Tokenization steps must be identical to steps for all documents (and possible queries)

Case Folding

- Reduce all letters to lower case
- Possible exceptions: capitalized words in mid-sentence
- MIT vs. mit
- Fed vs. fed
- It's often best to lowercase everything since users will use lowercase regardless of correct capitalization

Stop Words

- Stop words = extremely common words which have little value
- Examples: a, an, and, are, as, at, be, by, for, from, has, he, in, is, it, its, of, on, that, the, to, was, were, will, with
- Stop word elimination used to be standard in older IR systems
- But you need stop words for phrases, e.g., "King of Denmark"
- Nowadays NLP systems do not remove stopwords

Subword Tokenization

- Statistical model using a data-driven approach
- Trained in order to maximize the likelihood of a language model
- Used in many modern language applications for text understanding and generation
- Limited use for word embeddings

A modern solution for tokenization. Let's just learn to tokenize!



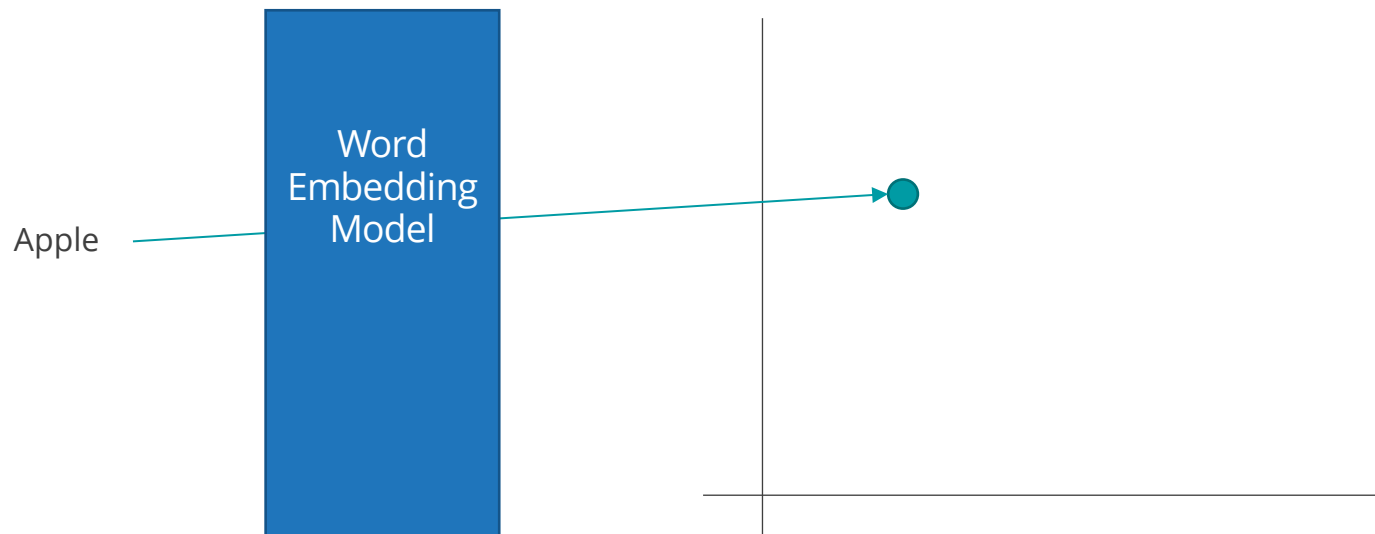
a	modern	solution	for	token	##ization	.	let	'	s	just	learn	to	token	##ize	!
---	--------	----------	-----	-------	-----------	---	-----	---	---	------	-------	----	-------	-------	---



Word Embeddings

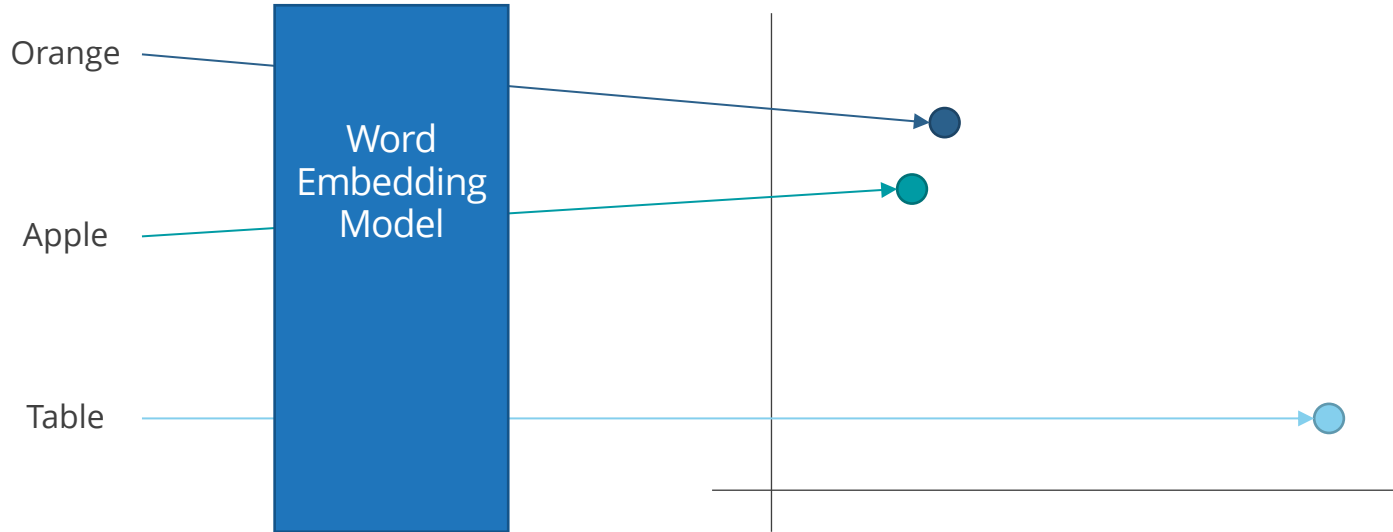
Goal of Embeddings

Semantic Representation of Words



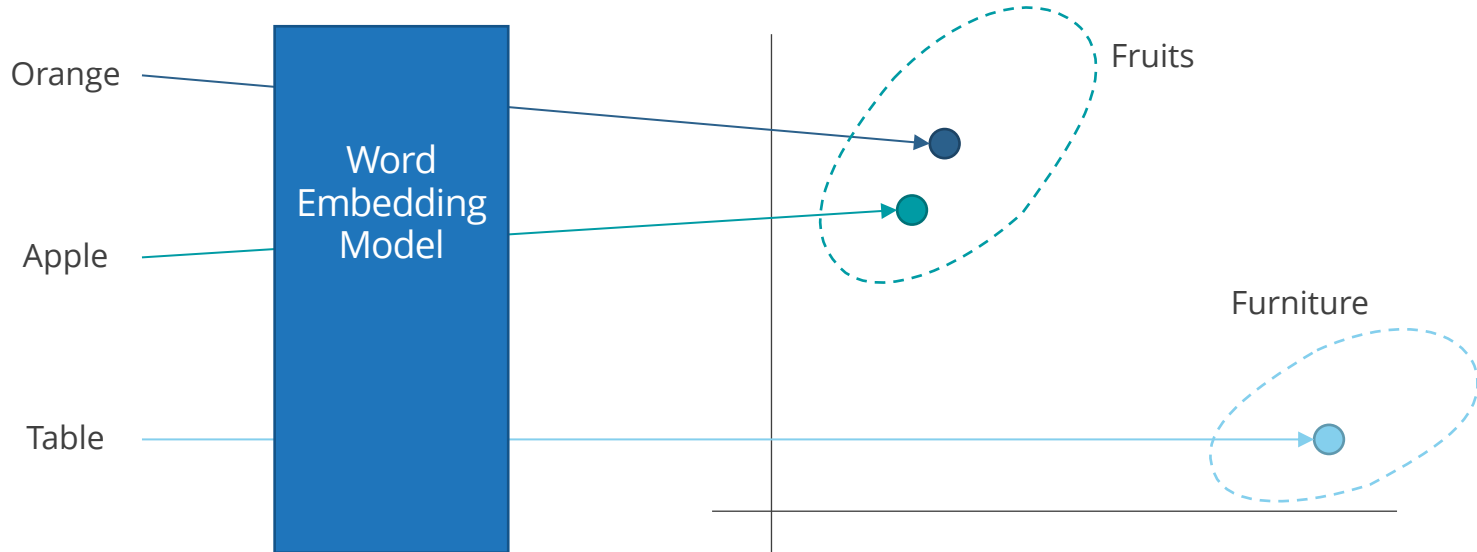
Goal of Embeddings

Semantic Representation of Words



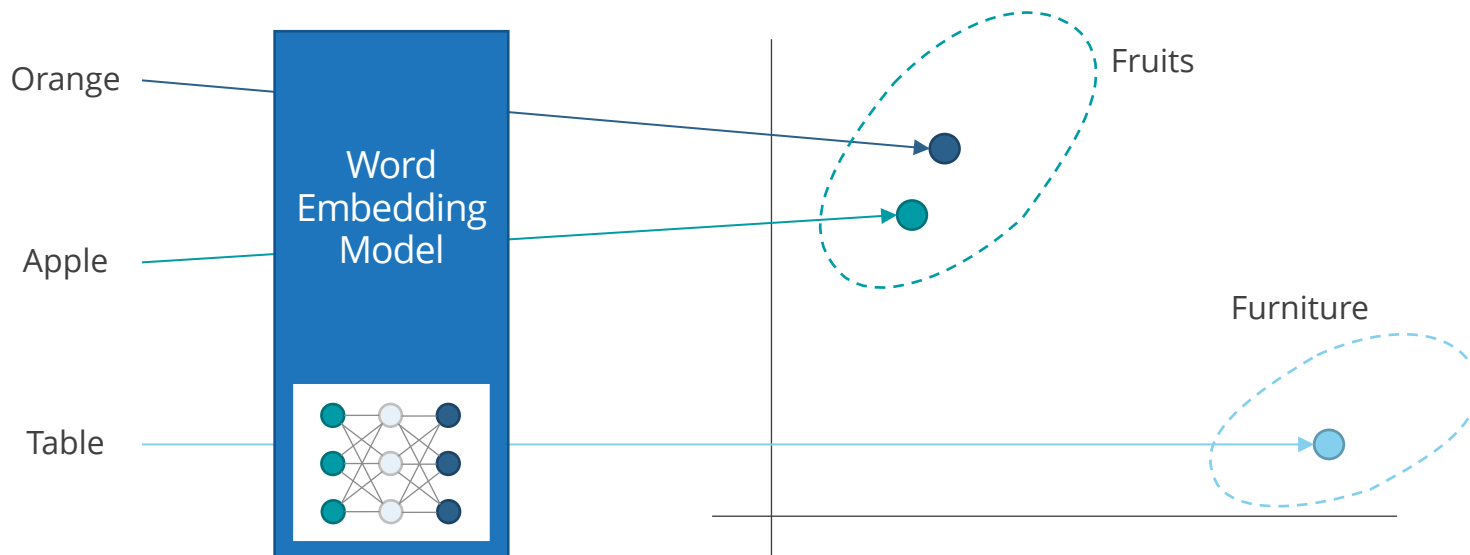
Goal of Embeddings

Semantic Representation of Words



Goal of Embeddings

Semantic Representation of Words



Vector Embedding of Words

Many embeddings methods developed in the recent years

- Latent Semantic Analysis/Indexing (1988)
 - Term weighting-based model
 - Consider occurrences of terms at document level
- Word2Vec (2013)
 - Prediction-based model
 - Consider occurrences of terms at context level
- GloVe (2014)
 - Count-based model
 - Consider occurrences of terms at context level
- ELMo (2018)
 - Language model-based
 - Contextual representations
- BERT (2018) / RoBERTa (2019) / ALBERT (2019)
 - Deeply bidirectional, unsupervised language representation
 - Contextual representations



Word2Vec

“A word is characterized by the company it keeps” (John Rupert Firth, 1957)

He was hit by a ??? bus.

Roses are ??? .

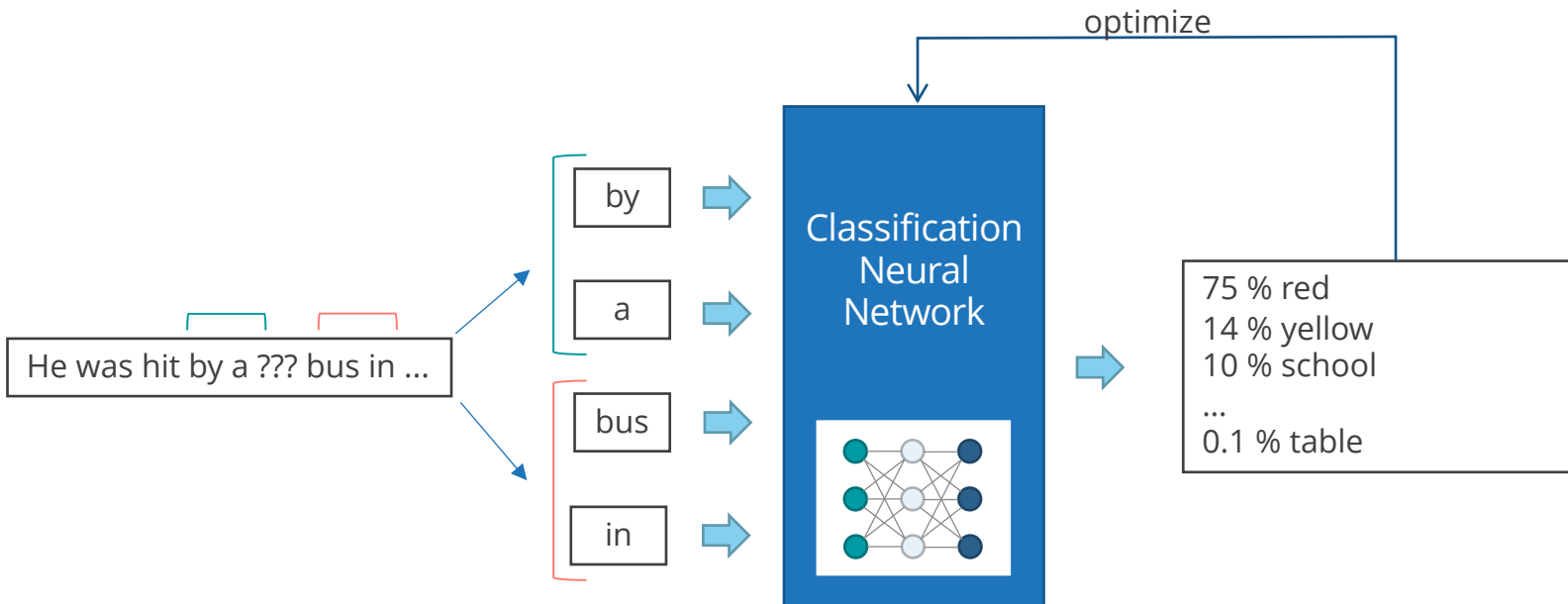
Question: Which word is ??? ?

She was holding a large, ??? balloon.

After being in the cold too long, her nose became ??? .

Idea

Ask a neural network the same question



... hit by a red bus in ...

Idea

- Train a classification algorithm to determine the probability that a **word w** appears close to a **target word**, e.g. “red”

We don't actually care about this task

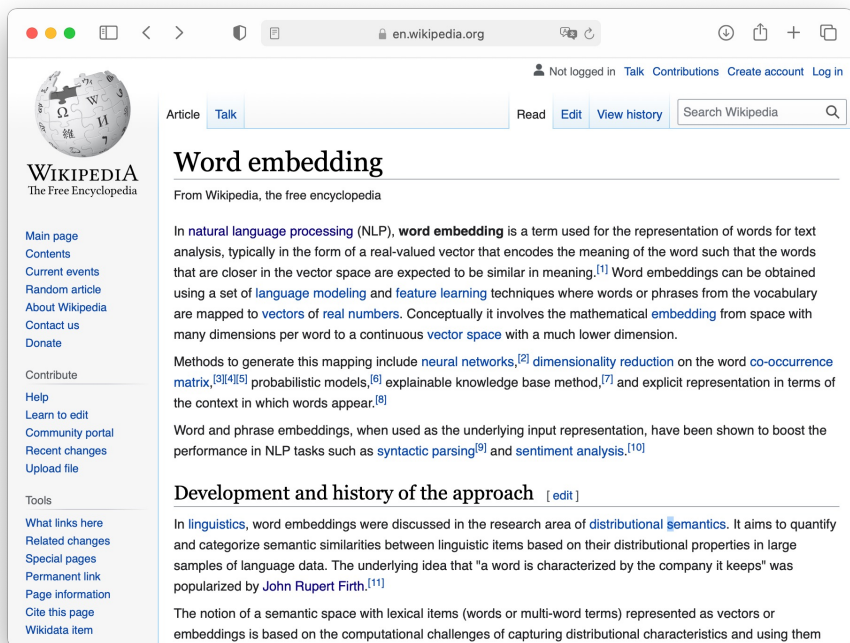
- But we'll take the learned classification weights as the word embeddings

Brilliant insight

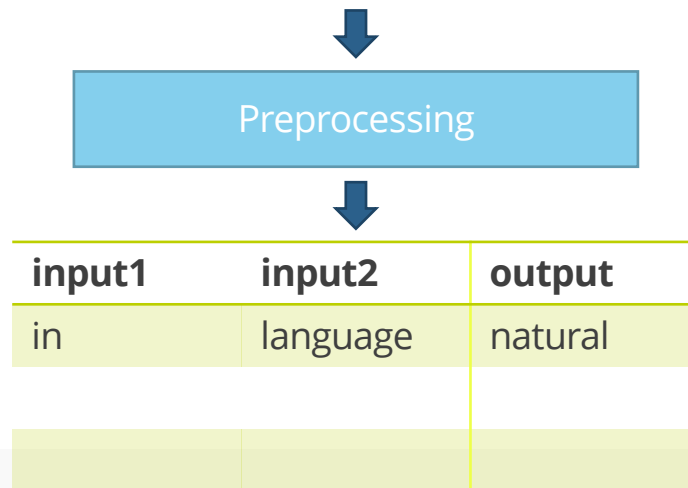
- Use running text as implicitly supervised training data!
- Generate gold ‘correct answer’ to the question “Is this target word likely to show up in this context?”

No need for hand-labeled supervision

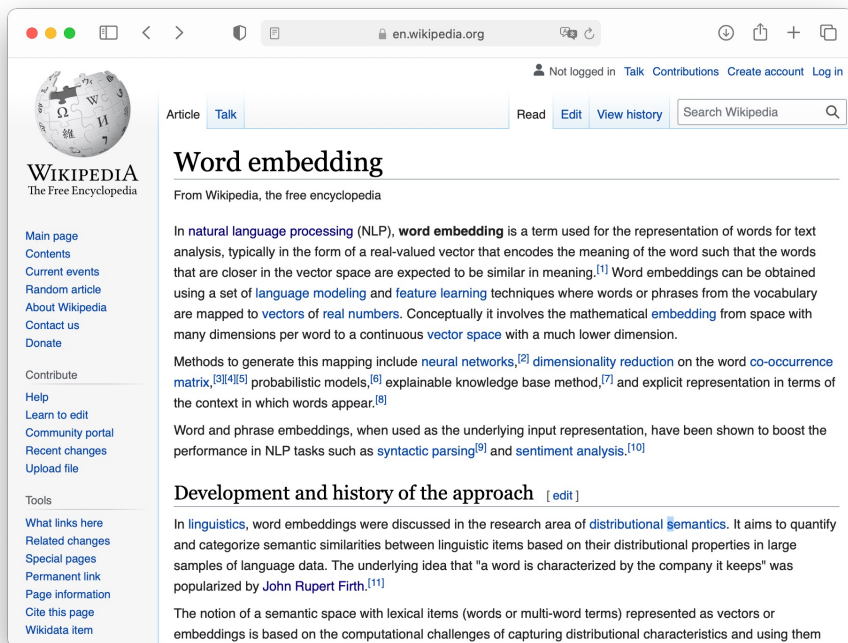
We have lot's of text for training in the internet.



➔ In natural language processing, word embedding is a term used for the representation of words for text analysis, typically in the form of a real-valued vector that encodes the meaning of the word such that the words that are closer in the vector space are expected to be similar in meaning.



We have lot's of text for training in the internet.



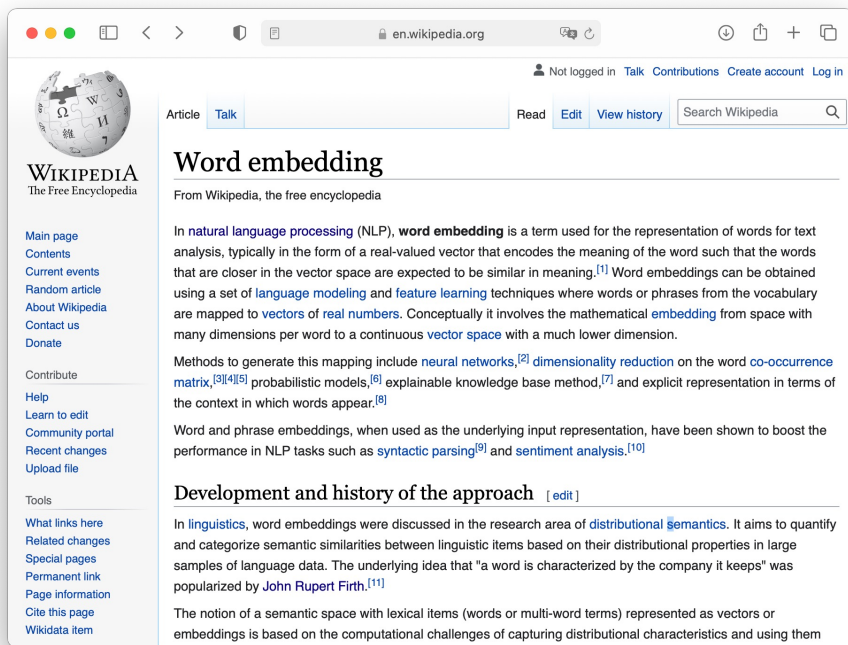
➔ In **natural language processing**, word embedding is a term used for the representation of words for text analysis, typically in the form of a real-valued vector that encodes the meaning of the word such that the words that are closer in the vector space are expected to be similar in meaning.

↓
Preprocessing

↓

input1	input2	output
in	language	natural
natural	processing	language

We have lot's of text for training in the internet.



In natural language processing, word embedding is a term used for the representation of words for text analysis, typically in the form of a real-valued vector that encodes the meaning of the word such that the words that are closer in the vector space are expected to be similar in meaning. ...

↓

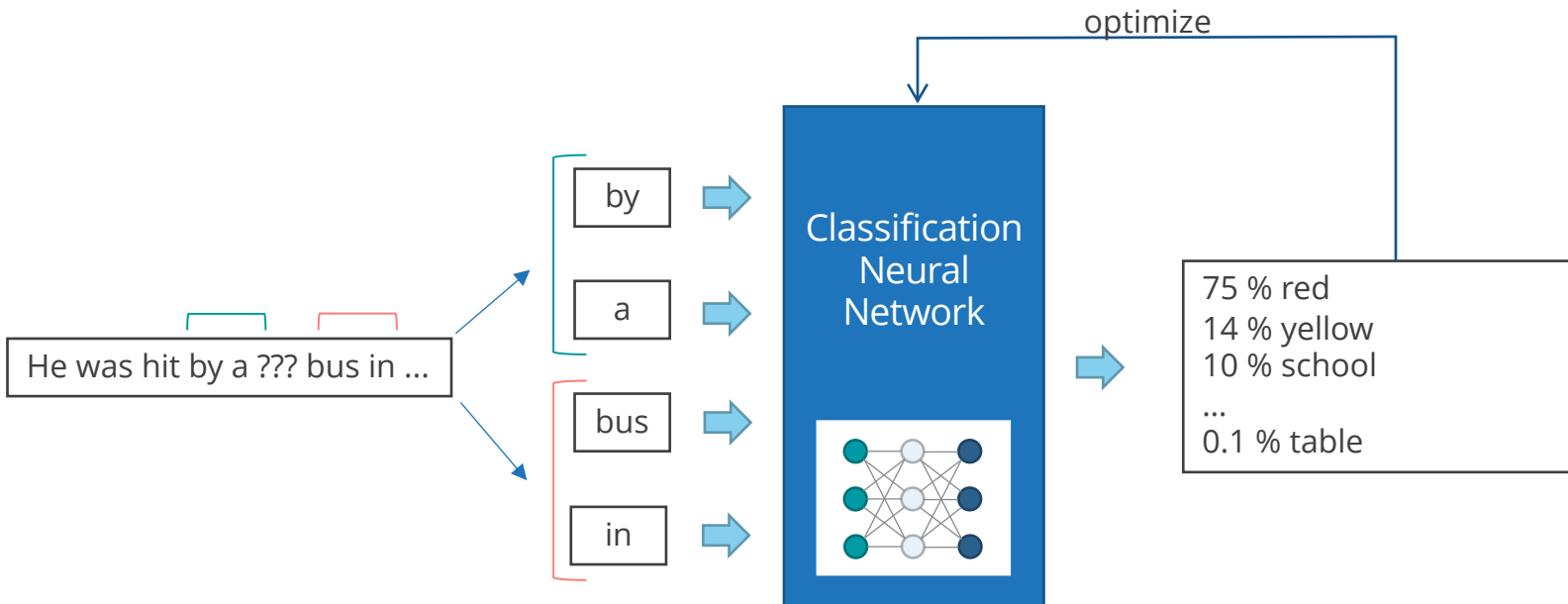
Preprocessing

↓

input1	input2	output
in	language	natural
natural	processing	language
language	word	processing

Training (naïve)

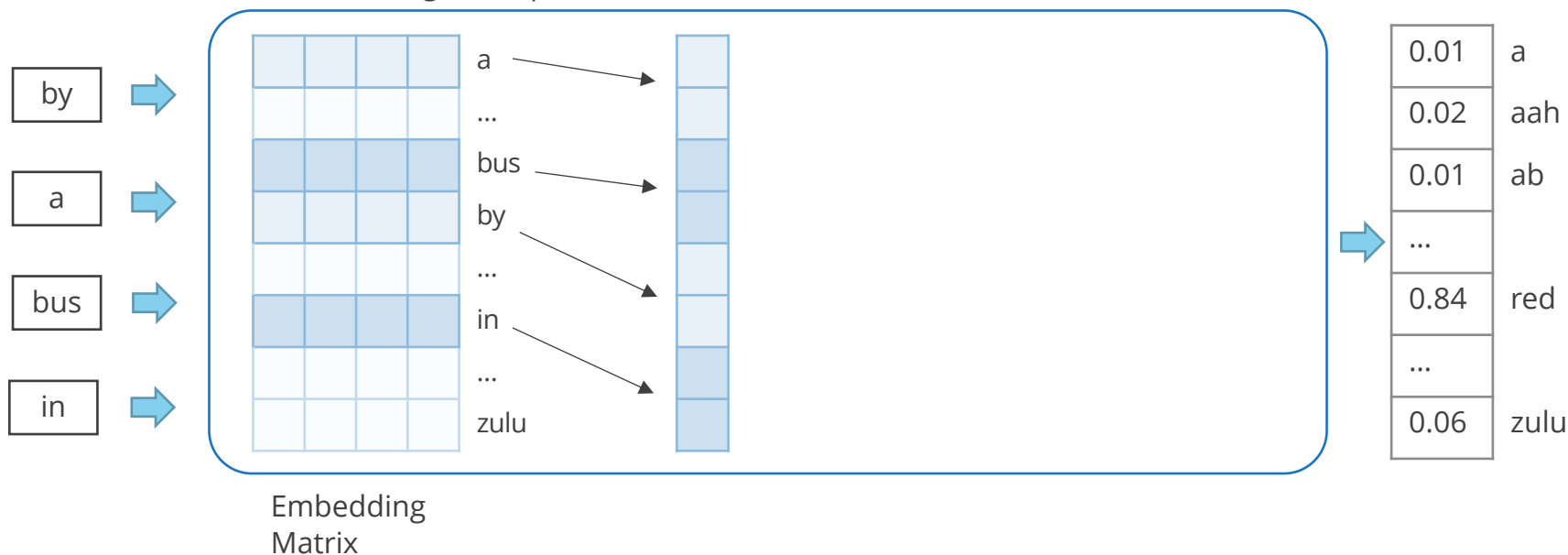
Predict the missing word



Training (naïve)

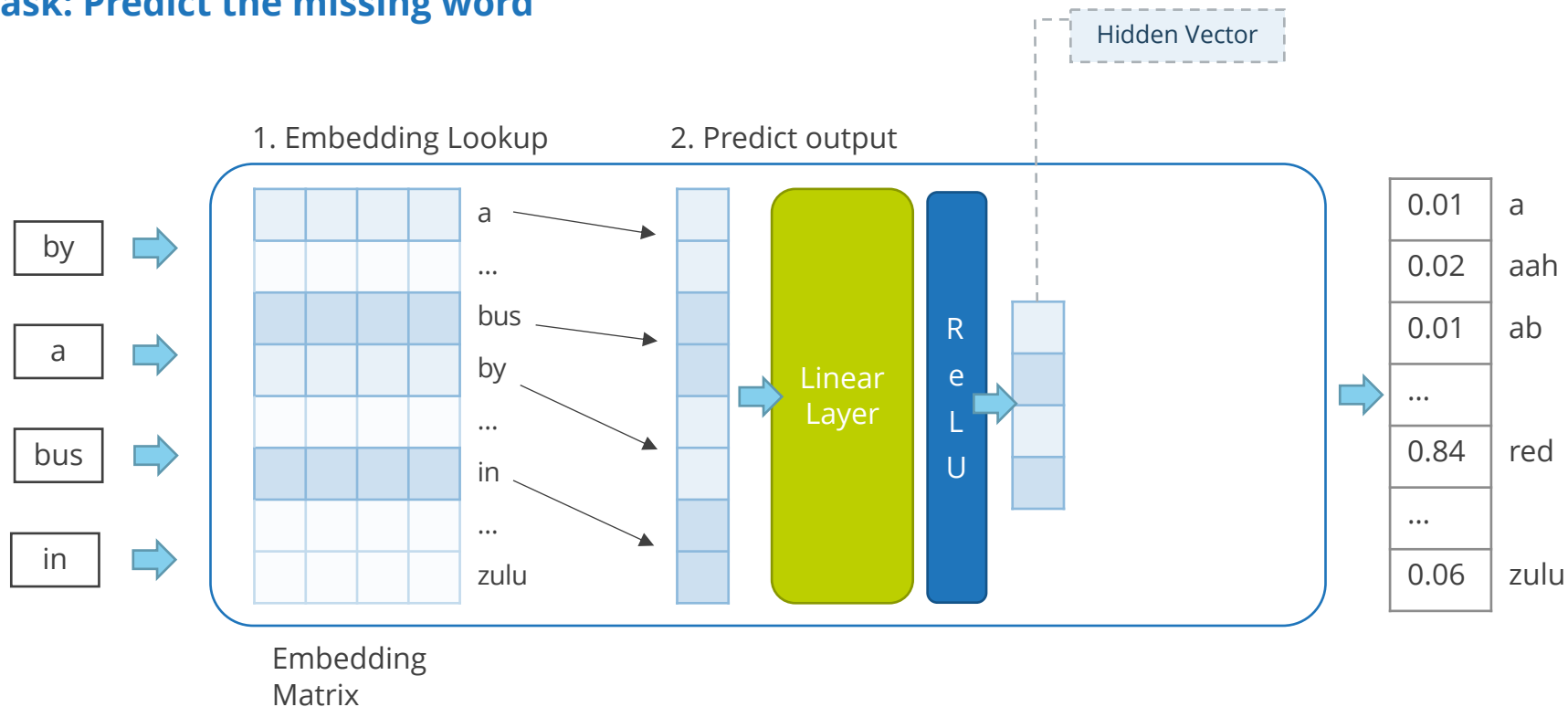
Task: Predict the missing word

1. Embedding Lookup



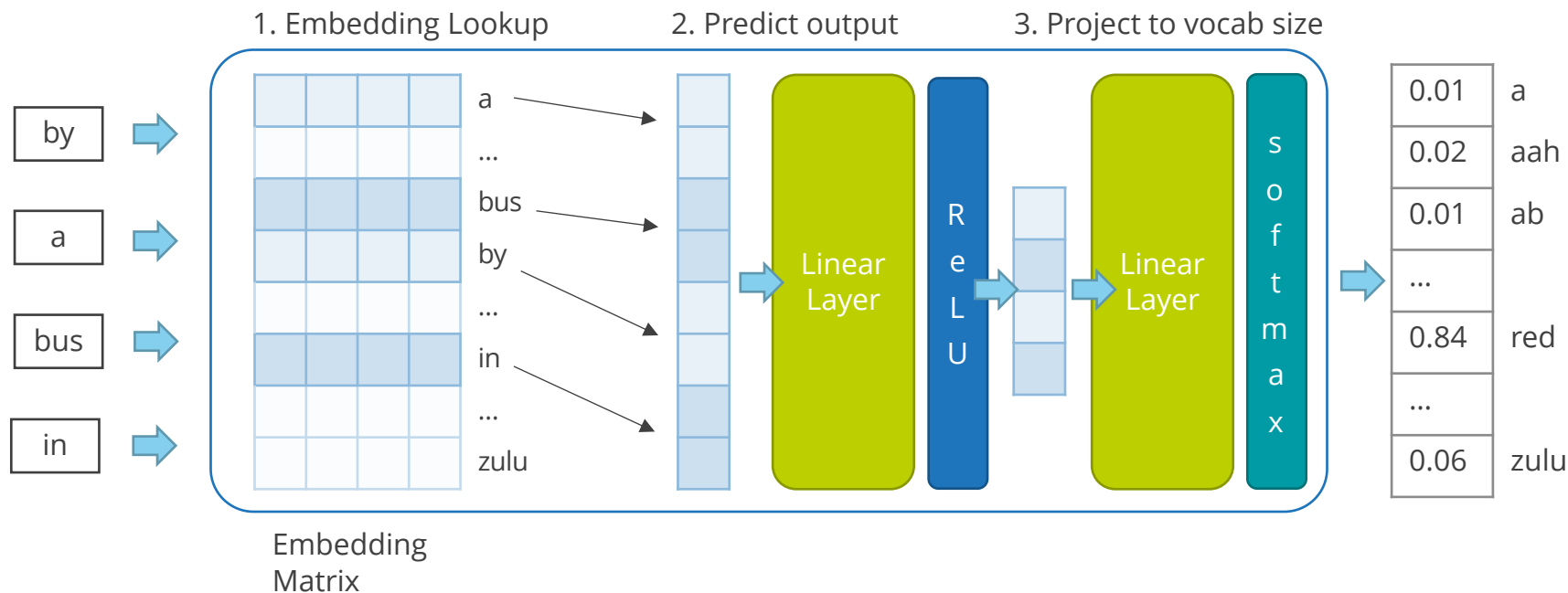
Training (naïve)

Task: Predict the missing word



Training (naïve)

Task: Predict the missing word

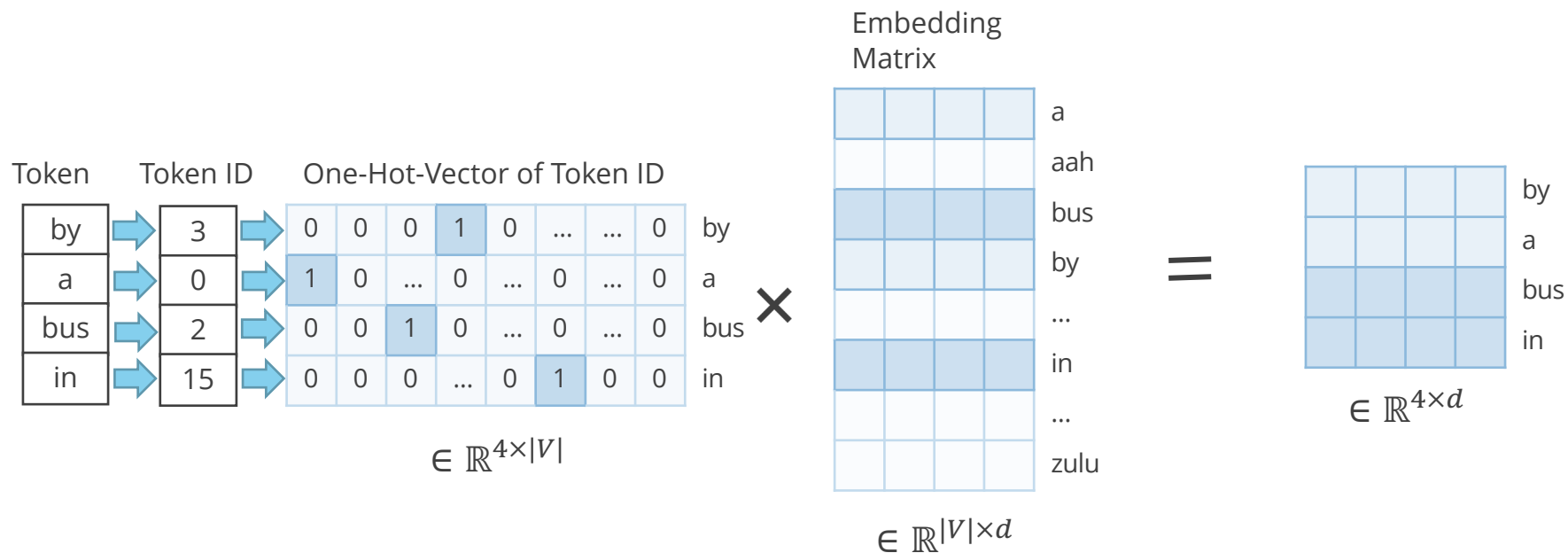


Embedding Lookup

Token	Token ID	One-Hot-Vector of Token ID	
by	3	0 0 0 1 0 0	by
a	0	1 0 ... 0 ... 0 ... 0	a
bus	2	0 0 1 0 ... 0 ... 0	bus
in	15	0 0 0 ... 0 1 0 0	in

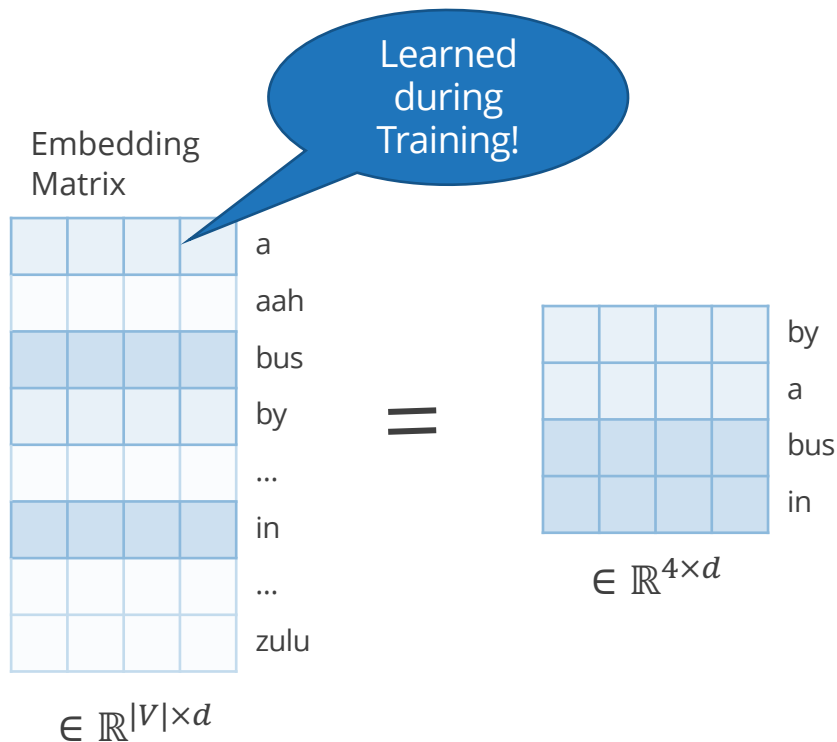
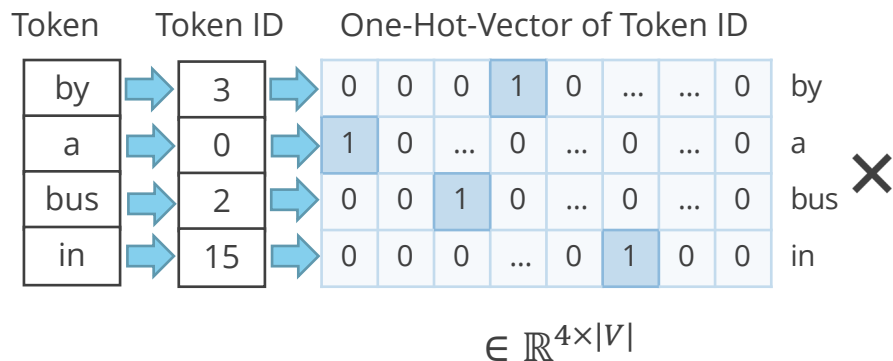
$$\in \mathbb{R}^{4 \times |V|}$$

Embedding Lookup



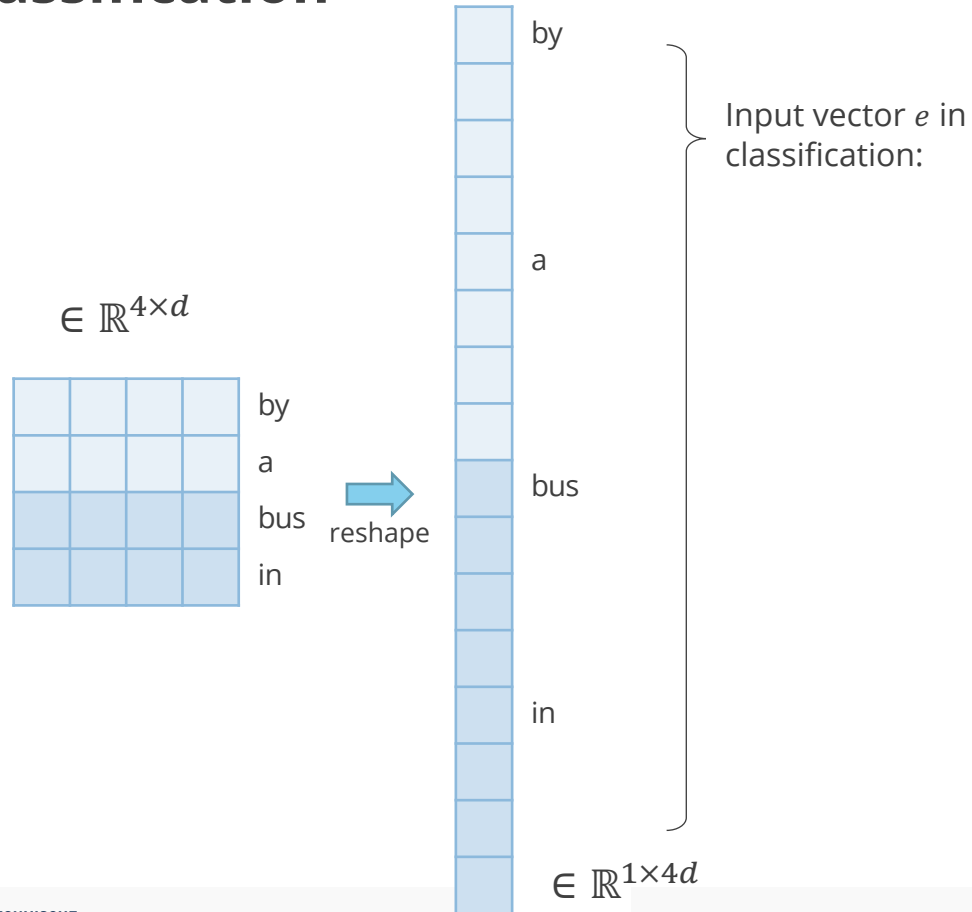
d ... embedding dimensionality, e.g., 100 – 300, here 4
 $|V|$... size of vocabulary V

Embedding Lookup

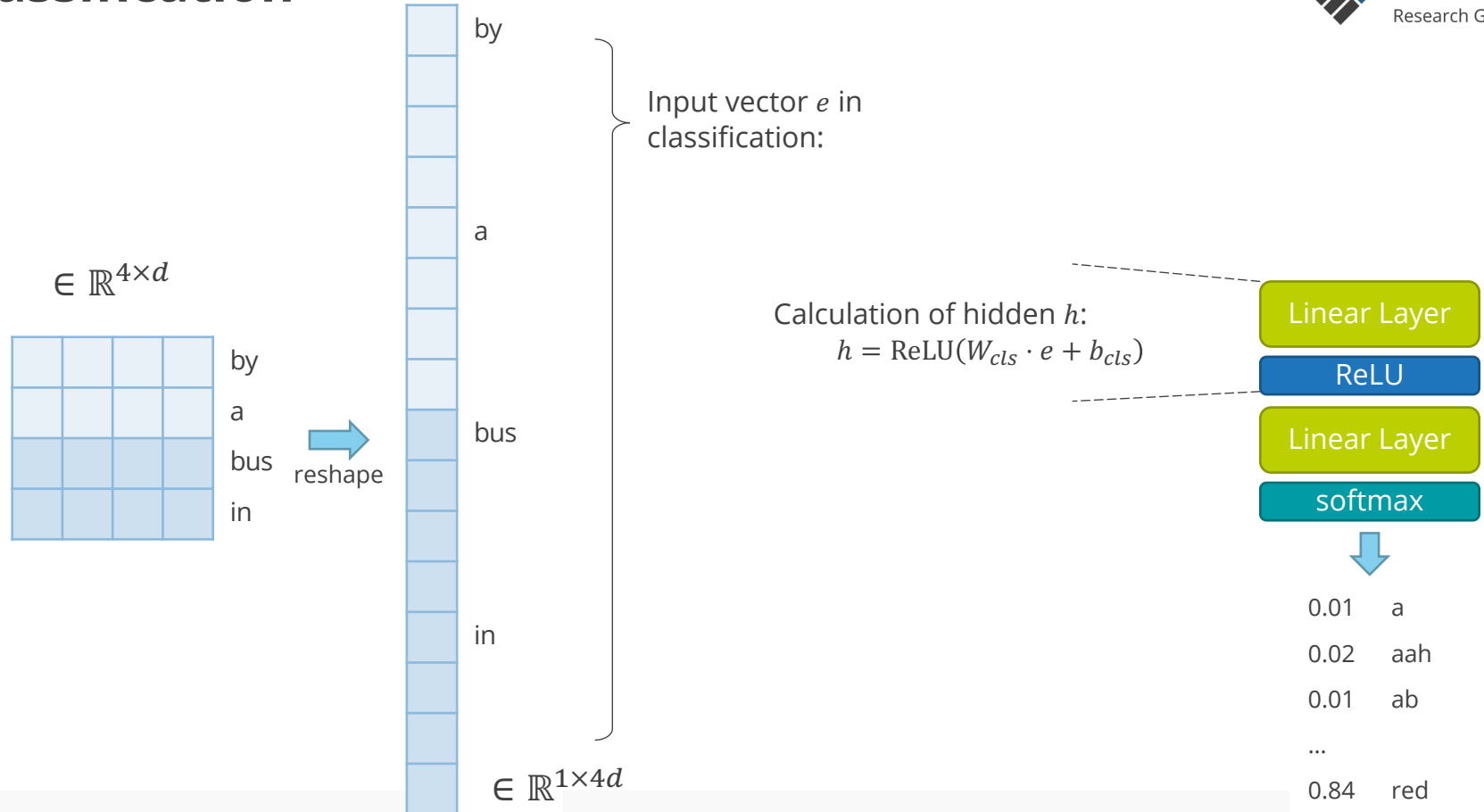


d ... embedding dimensionality, e.g., 100 – 300, here 4
 $|V|$... size of vocabulary V

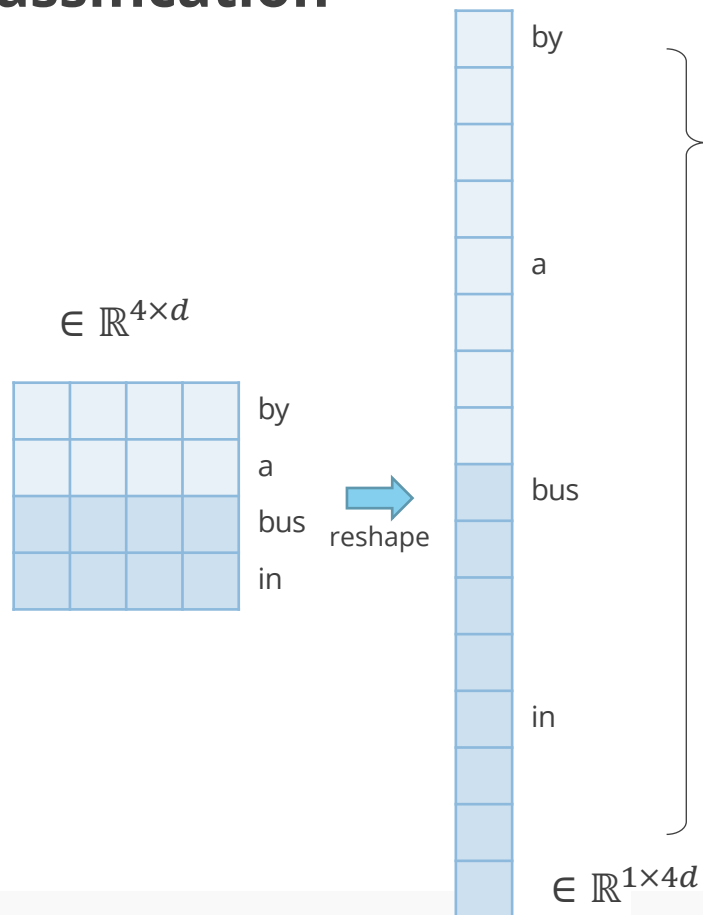
Classification



Classification



Classification



Input vector e in classification:

Projection to vocabulary:
 $o = W_{proj} \cdot h + b_{proj}$

Linear Layer

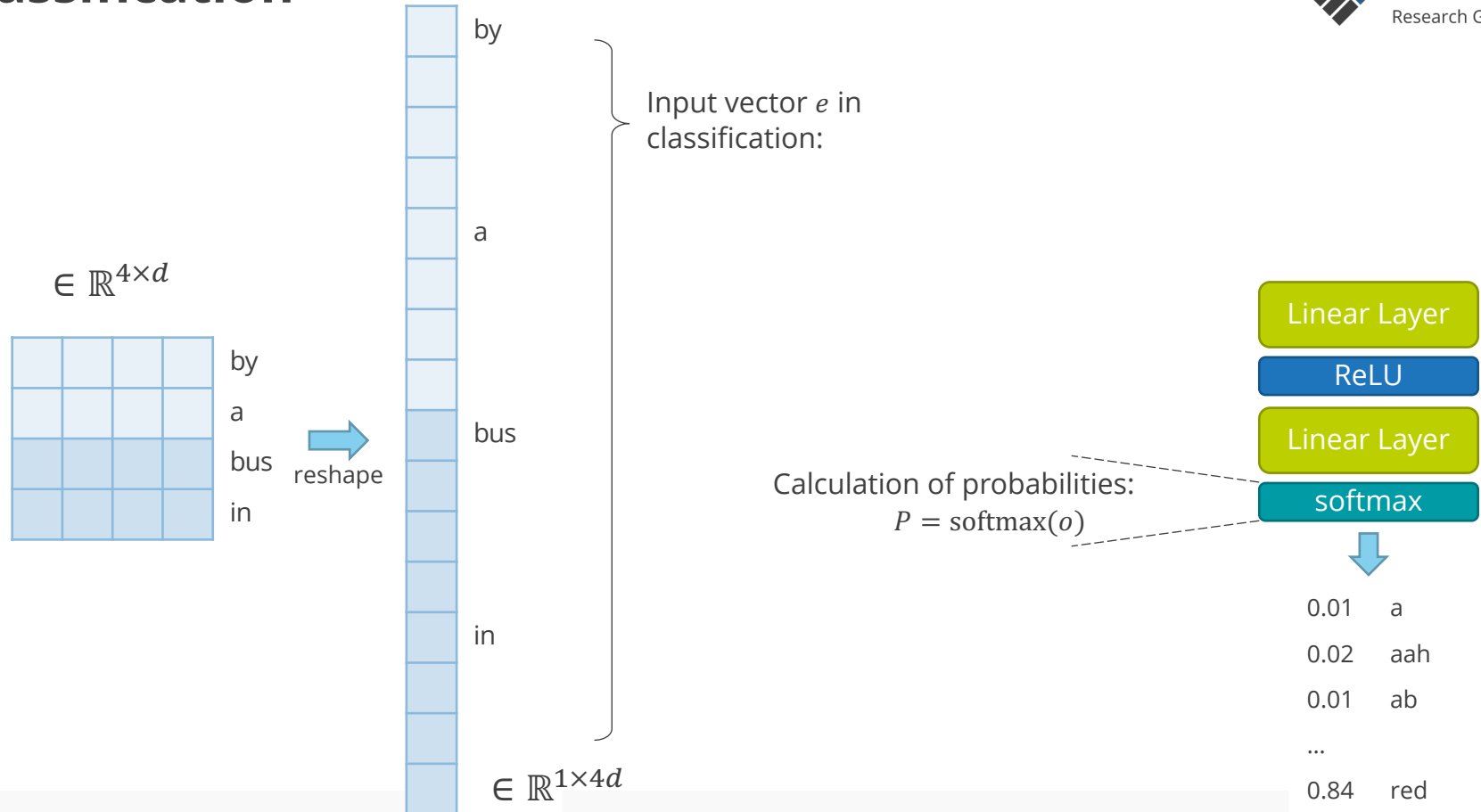
ReLU

Linear Layer

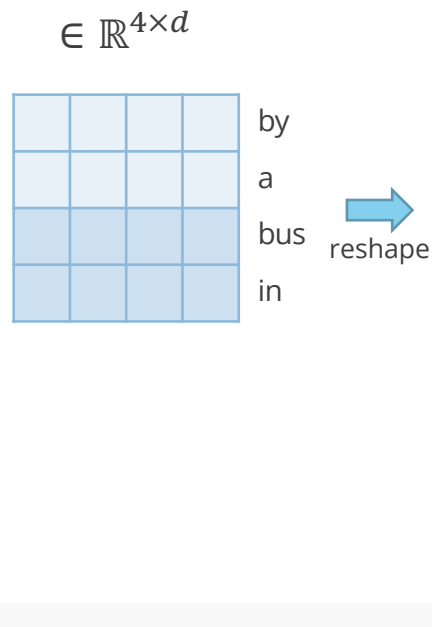
softmax

0.01 a
0.02 aah
0.01 ab
...
0.84 red

Classification



Classification



Input vector e in classification:

Calculation of hidden h :

$$h = \text{ReLU}(W_{cls} \cdot e + b_{cls})$$

Projection to vocabulary:

$$o = W_{proj} \cdot h + b_{proj}$$

Calculation of probabilities:

$$P = \text{softmax}(o)$$

with

$$W_{cls} \in \mathbb{R}^{4d \times d_{hidden}},$$

$$W_{proj} \in \mathbb{R}^{d_{hidden} \times |V|}.$$

d_{hidden} is typically smaller than d , e.g., 128

Linear Layer

ReLU

Linear Layer

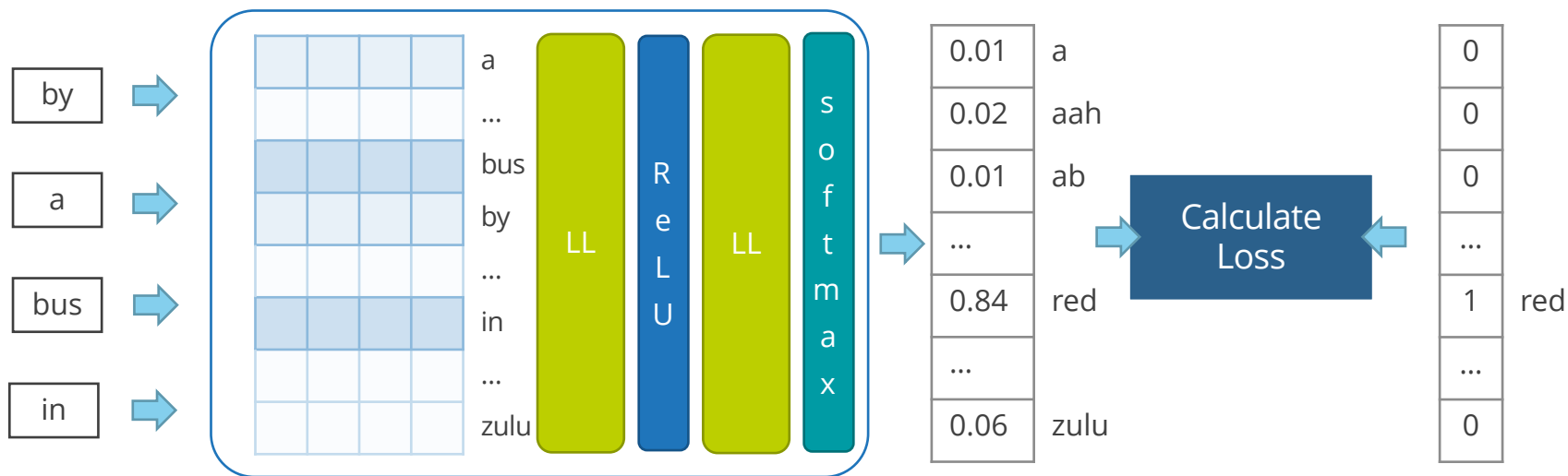
softmax



0.01	a
0.02	aah
0.01	ab
...	
0.84	red

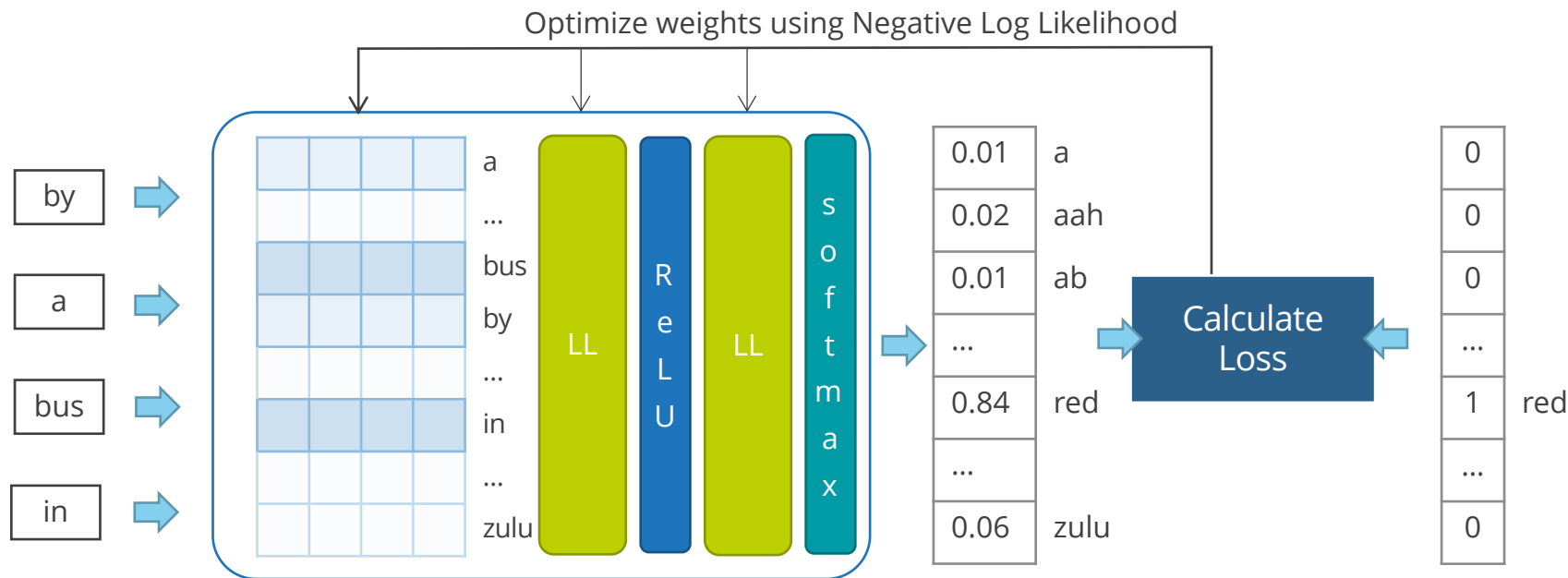
Training (naïve)

Compare to true label



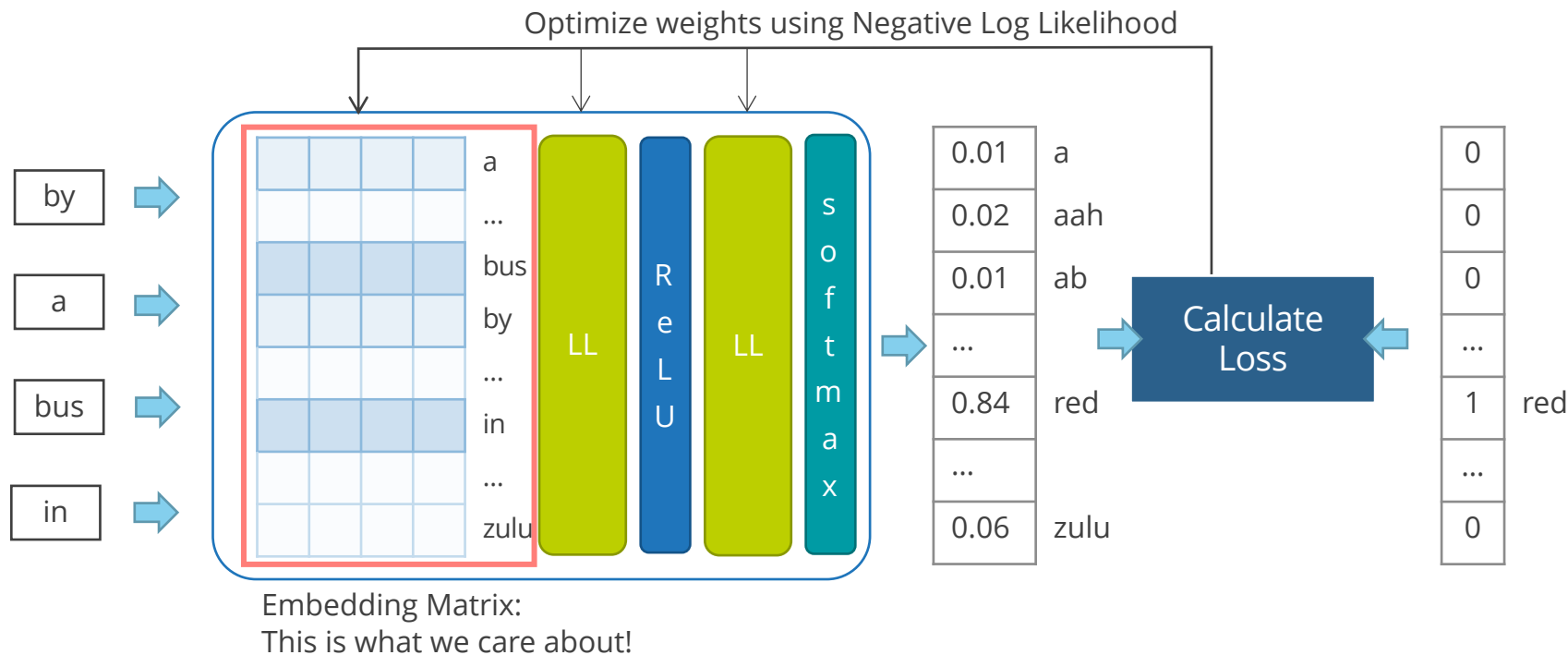
Training (naïve)

Compare to true label



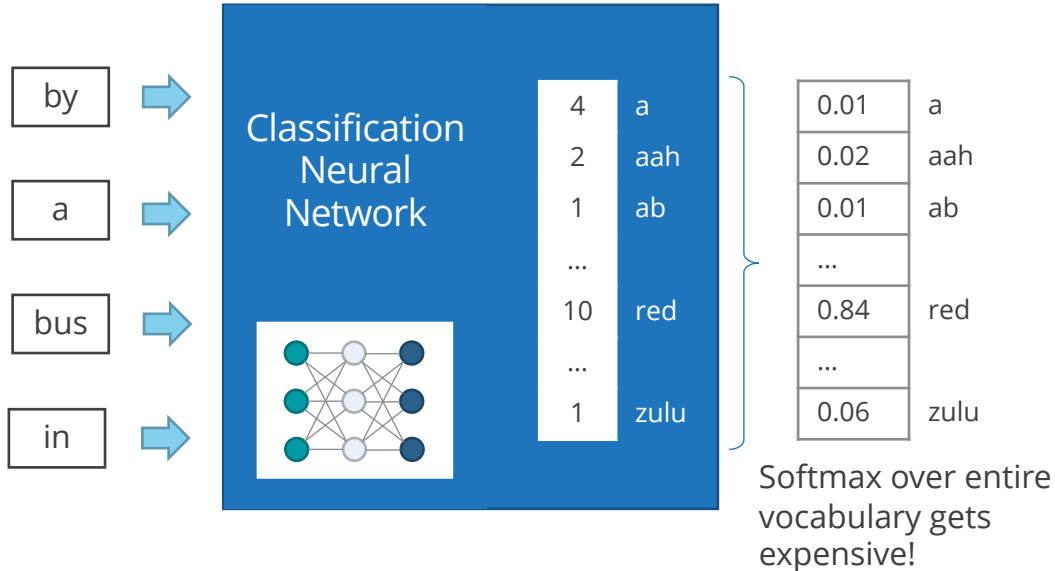
Training (naïve)

Compare to true label



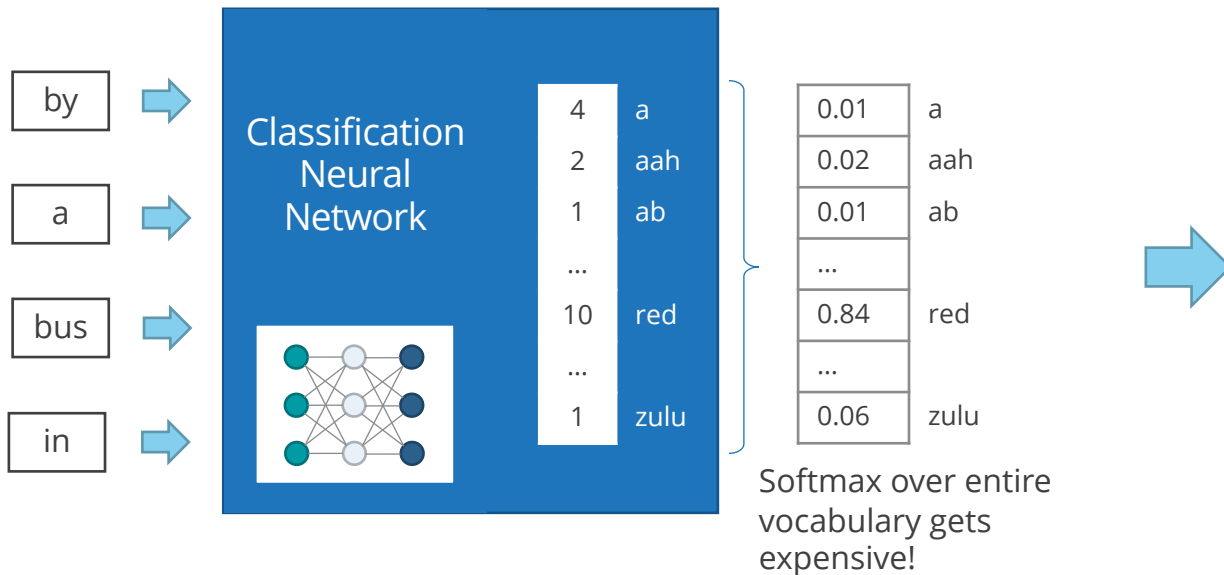
Modify the task

Softmax of vocabulary gets expensive!



Modify the task

Softmax of vocabulary gets expensive!

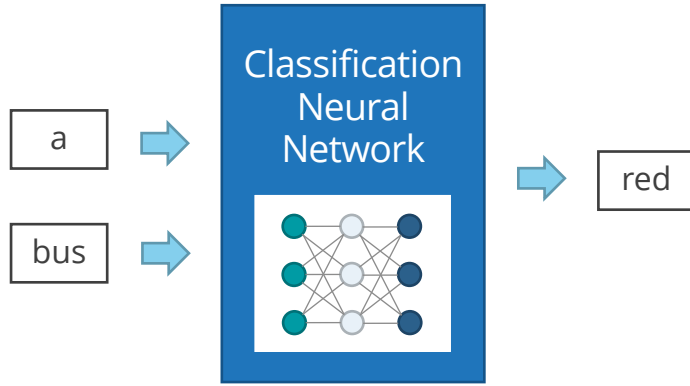


Instead of:
Predict the missing word.
→ Thousands of classes

Classify:
Will two words appear
together?
→ Binary Classification

Modify the task

So far

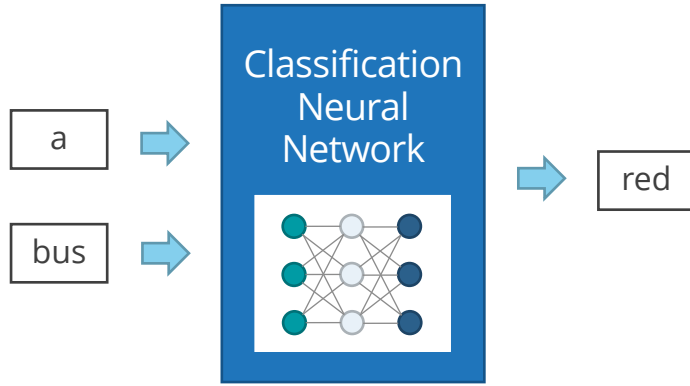


Data

input1	input2	output
by	red	a
a	bus	red
red	in	bus

Modify the task

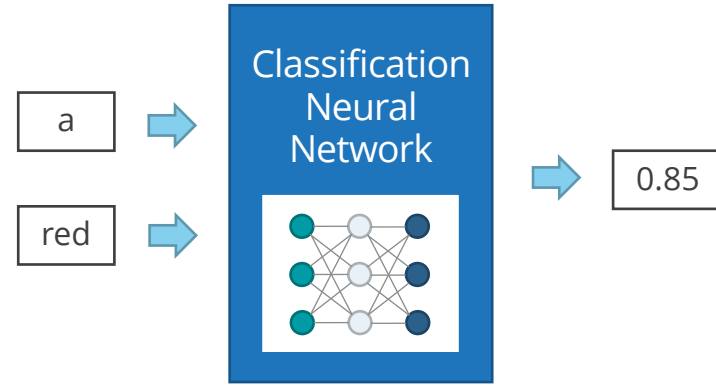
So far



Data

input1	input2	output
by	red	a
a	bus	red
red	in	bus

Afterwards

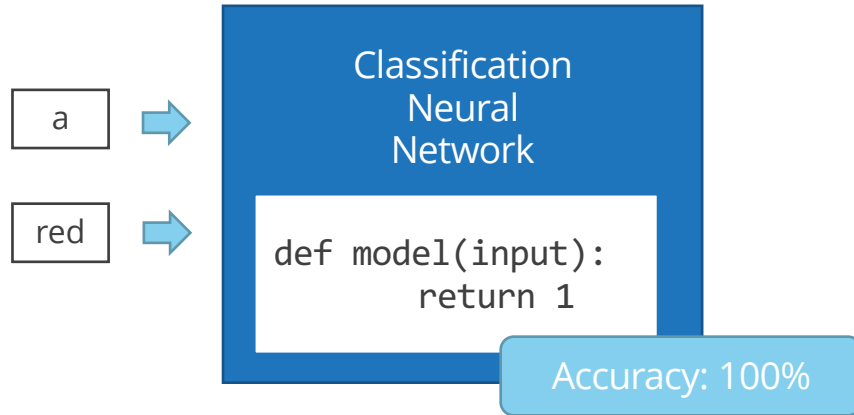


input1	input2	output
a	red	1
red	bus	1
bus	in	1

Binary
output:
0 = No
1 = Yes

Modify the task

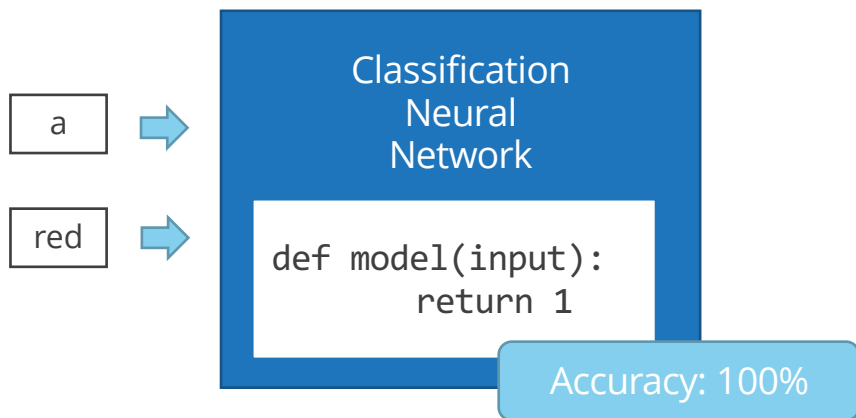
What happens with this model?



input1	input2	output
a	red	1
red	bus	1
bus	in	1

Modify the task

What happens with this model?



input1	input2	output
a	red	1
red	bus	1
bus	in	1

Negative Sampling

→ Sample random words that are not neighbours

input1	input2	output
a	red	1
a	running	0
red	bus	1
red	hide	0
bus	in	1
bus	fire	0

Binary
output:
0 = No
1 = Yes

Two variants to construct this task

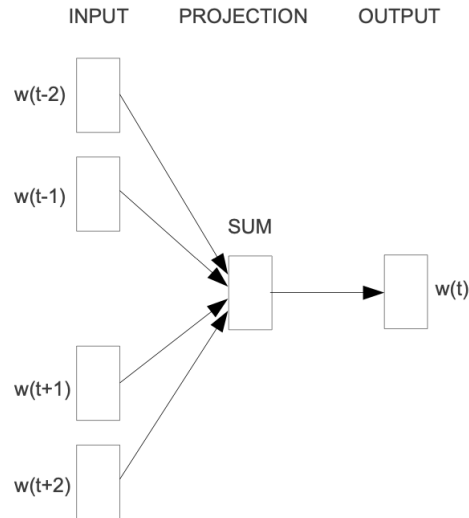
Continuous bag of words (CBOW)

- Predict word from context

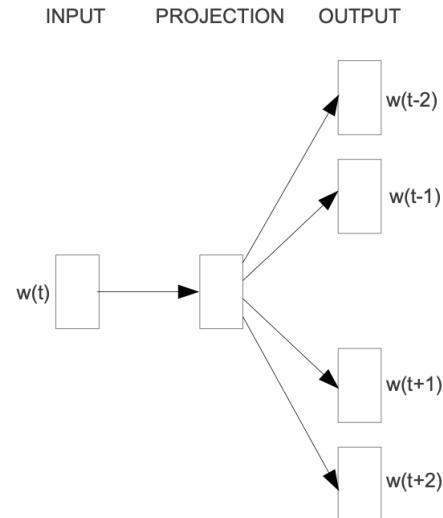
Skipgram

- Predict context words from word

Both variants can be used for the naïve approach and the optimized binary classification approach.



CBOW



Skip-gram

Predict context words from target word

by a red bus in



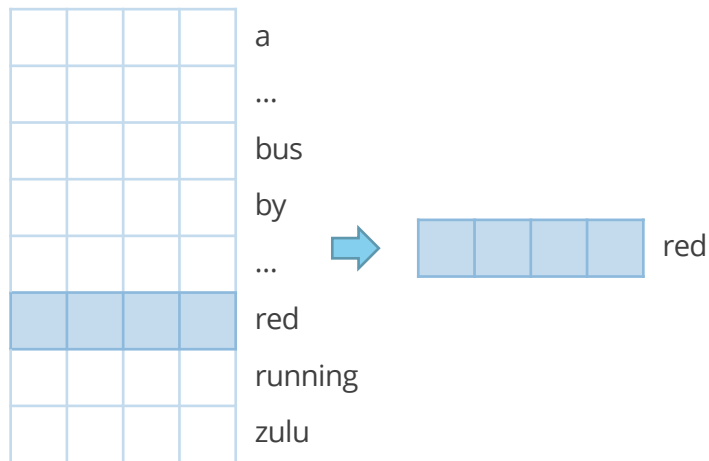
Fixed
target
word

target	context	output
red	by	1
red	running	0
red	a	1
red	hide	0
red	bus	1
red	fire	0
red	in	1
red	cup	0

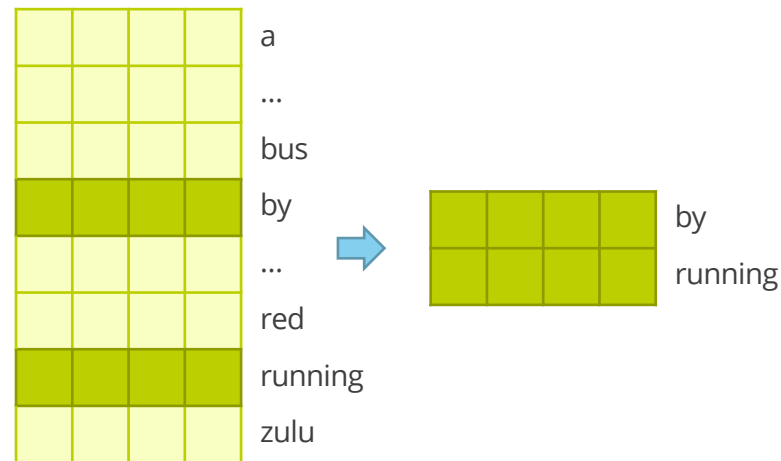
Skipgram

Look up Embeddings

target	context	output
red	by	1
red	running	0



Embedding Matrix

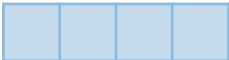
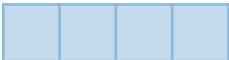


Context Matrix

Skipgram

Calculate output

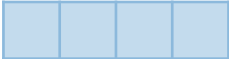
target	context	output
red	by	1
red	running	0

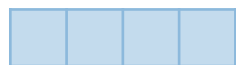
Target Word t	Context Word c	$t \cdot c$	$\text{sigmoid}(t \cdot c)$
 red	 by	0.51	0.66
 red	 running	-1.10	0.25

Skipgram

Calculate output

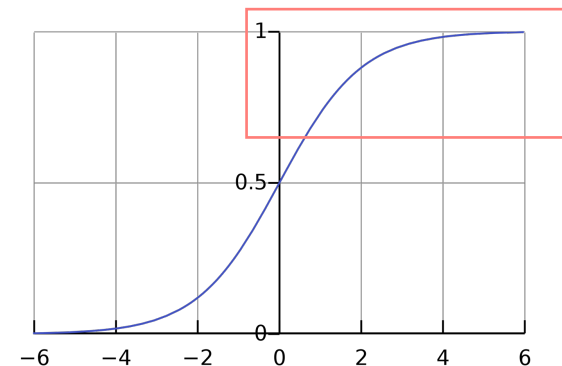
target	context	output
red	by	1
red	running	0

Target Word t	Context Word c	$t \cdot c$	$\text{sigmoid}(t \cdot c)$
 red	 by	0.51	0.66



red

We want: $\text{sigmoid}(t \cdot c) = 1$
 $t \cdot c$ needs to be as large as possible $\rightarrow t \cdot c = 1$
That tells us: $t = c$, vectors in same direction

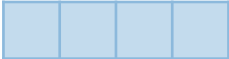


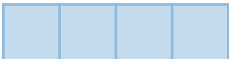
Skipgram

Calculate output

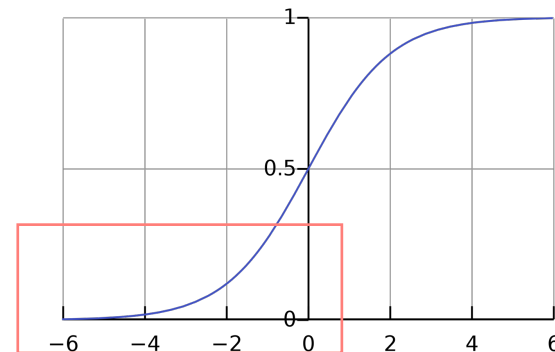
target	context	output
red	by	1
red	running	0

Target Word t	Context Word c	$t \cdot c$	$\text{sigmoid}(t \cdot c)$
-----------------	------------------	-------------	-----------------------------

 red	 by	0.51	0.66
---	--	------	------

 red	 running	-1.10	0.25
---	---	-------	------

We want: $\text{sigmoid}(t \cdot c) = 0$
 $t \cdot c$ needs to be as small as possible $\rightarrow t \cdot c = -1$
That tells us: $t = -c$, vectors in different direction

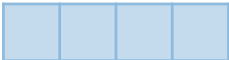
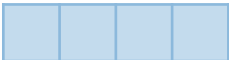


Skipgram

Calculate output

target	context	output
red	by	1
red	running	0

Error = output - $\text{sigmoid}(t \cdot c)$

Target Word t	Context Word c	$t \cdot c$	$\text{sigmoid}(t \cdot c)$	Error
 red	 by	0.51	0.66	0.34
 red	 running	-1.10	0.25	-0.25

Used to update the embedding matrices

Summary Word2Vec

Idea: Words are defined by their context

- **First: Naïve Algorithm for CBOW**
→ Softmax over Vocabulary is expensive!
- **Therefore: Change the task to binary classification + Negative Sampling**
- **Word Embeddings are trained as model parameters**
- **Lot's of training data available on the internet**



Summary Word2Vec

Idea: Words are defined by their context

- **First: Naïve Algorithm for CBOW**

→ Softmax over Vocabulary is expensive!

- **Therefore: Change the task to binary classification + Negative Sampling**

- **Word Embeddings are trained as model parameters**

- **Lot's of training data available on the internet**

Hyperparameters:

- Window Size
- Algorithm: CBOW vs. Skipgram
- Number of negative samples

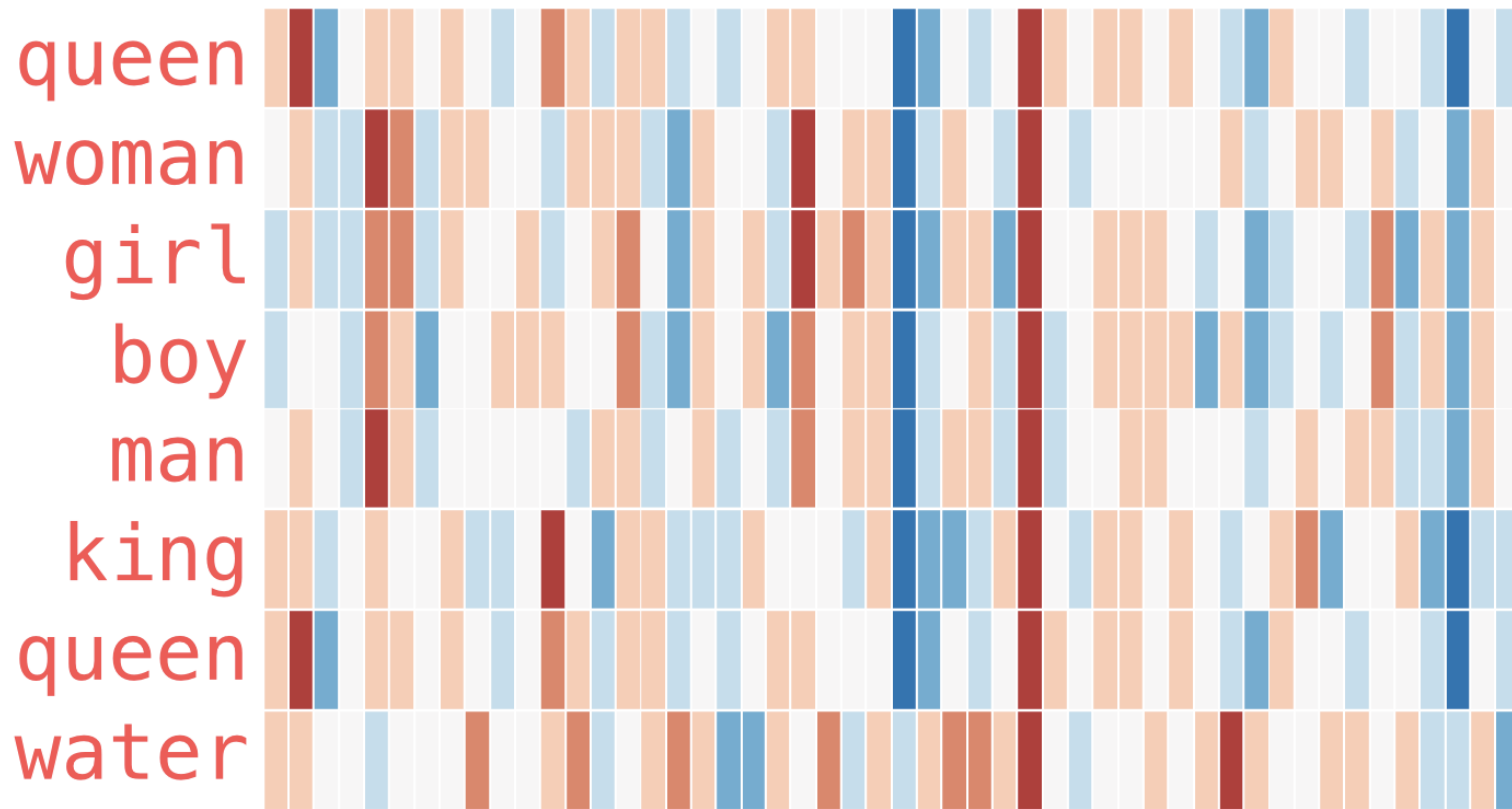
... influence the quality of the embeddings!





Applications

Trained Embeddings



Cosine-Similarity

Most similar words to 'Pizza':

Nearest points in the original space:

pasta

sauce

bagel

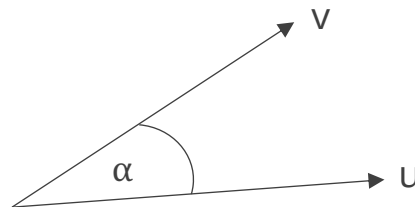
dishes

funghi

chicken

dish

macaroni



Dot-Product of vectors u and v :

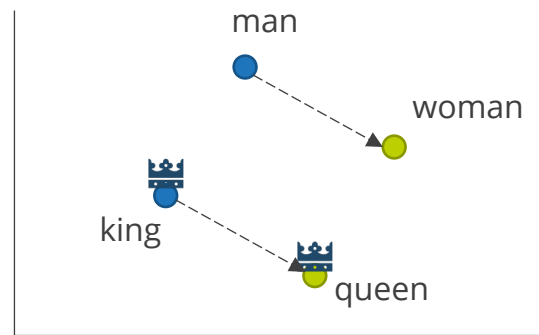
$$u \cdot v = |u| \cdot |v| \cdot \cos \alpha$$

Cosine-Similarity of vector u and v :

$$\text{Sim}(u, v) := \cos(\alpha) = \frac{u \cdot v}{|u| \cdot |v|} = \frac{\sum_i u_i \cdot v_i}{\sqrt{\sum_i u_i^2} \cdot \sqrt{\sum_i v_i^2}}$$

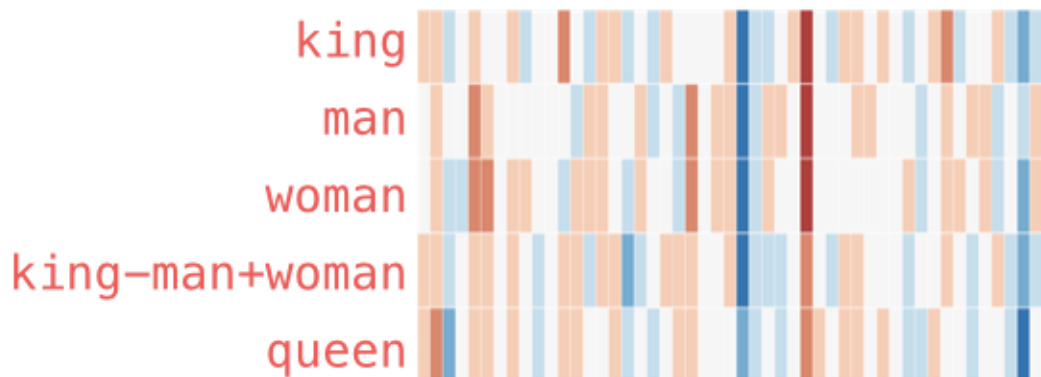
<http://projector.tensorflow.org>

Analogy



Idea: $\text{man} - \text{woman} = \text{king} - \text{queen}$

$\text{king} - \text{man} + \text{woman} \approx \text{queen}$



Newspapers			
New York San Jose	New York Times San Jose Mercury News	Baltimore Cincinnati	Baltimore Sun Cincinnati Enquirer
NHL Teams			
Boston Phoenix	Boston Bruins Phoenix Coyotes	Montreal Nashville	Montreal Canadiens Nashville Predators
NBA Teams			
Detroit Oakland	Detroit Pistons Golden State Warriors	Toronto Memphis	Toronto Raptors Memphis Grizzlies
Airlines			
Austria Belgium	Austrian Airlines Brussels Airlines	Spain Greece	Spainair Aegean Airlines
Company executives			
Steve Ballmer Samuel J. Palmisano	Microsoft IBM	Larry Page Werner Vogels	Google Amazon

Table 2: Examples of the analogical reasoning task for phrases (the full test set has 3218 examples). The goal is to compute the fourth phrase using the first three. Our best model achieved an accuracy of 72% on this dataset.

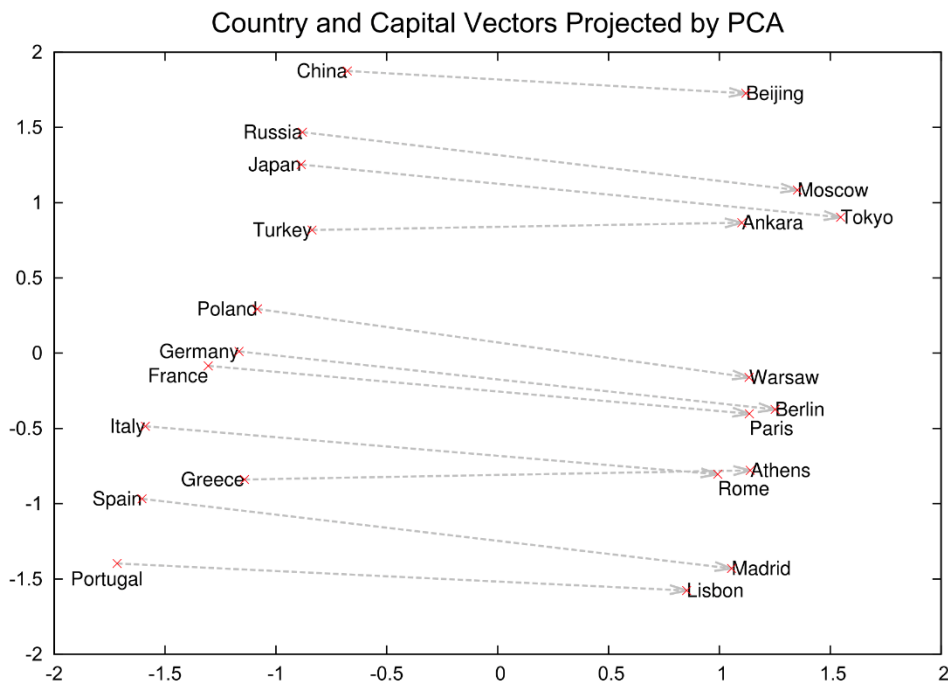


Figure 2: Two-dimensional PCA projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities. The figure illustrates ability of the model to automatically organize concepts and learn implicitly the relationships between them, as during the training we did not provide any supervised information about what a capital city means.

Word Embeddings are used as Input for Neural Networks

- Words are discrete, but neural networks need numeric input
- Embeddings can also be used for other types of categorical (non-numerical) input

Applications

- Information Retrieval
- Text understanding
- Classification
- Part-of-Speech Tagging
- Named-Entity-Recognition
- Much more ...

Limitations

Training Bias

Extreme *she*

1. homemaker
2. nurse
3. receptionist
4. librarian
5. socialite
6. hairdresser
7. nanny
8. bookkeeper
9. stylist
10. housekeeper

Extreme *he*

1. maestro
2. skipper
3. protege
4. philosopher
5. captain
6. architect
7. financier
8. warrior
9. broadcaster
10. magician

No Context

- The **finish** of the race
- This is a **Finish** dish.

→ Same word = Same vector

No Word Order

- This is **good**, it's not **bad**.
- This is **bad**, it's not **good**.

→ Same words = Same Vectors

Advances of Word2Vec

- Not just words, but parts of words: FastText (good for compound words)
- Context-dependend embeddings (Think of Polish vs. to polish): BERT, ELMO

Related Ideas

- Sentence2Vec, Doc2Vec
- Embeddings of graphs, images, larger texts, ...
- Recommendations

Sources and further readings

- <http://jalammar.github.io/illustrated-word2vec/>
- **Word2Vec Papers (Mikolov, 2013):**
 - <https://arxiv.org/pdf/1301.3781.pdf>
 - <https://arxiv.org/pdf/1310.4546.pdf>
- <http://projector.tensorflow.org>
- <https://radimrehurek.com/gensim/>
- **More Details and Derivatives:** <https://www.youtube.com/watch?v=ERibwqs9p38>

Next Exercise: We will use Google Colab!

PyTorch Lösung.ipynb

File Edit View Insert Runtime Tools Help Last edited on Jan 18, 2022

+ Code + Text

Let's visualize Tensors!

```
[ ] data = [[[0, 1, 2, 3, 4], [5, 6, 7, 8, 9]],
             [[10, 11, 12, 13, 14], [15, 16, 17, 18, 19]],
             [[20, 21, 22, 23, 24], [25, 26, 27, 28, 29]]]
x = torch.tensor(data)
x
```

tensor([[[[0, 1, 2, 3, 4],
 [5, 6, 7, 8, 9]],
 [[10, 11, 12, 13, 14],
 [15, 16, 17, 18, 19]],
 [[20, 21, 22, 23, 24],
 [25, 26, 27, 28, 29]]]])

<> Exercise 1

Reshape the 3D Tensor x into 3 different 2D Tensor.

The first one should have the shape (3, 5), the second one should have the shape (2, 5) and the third one should have the shape (3, 2). The

Anja Reusch

Dresden Database Research Group

12. Text Processing

Scalable Data Engineering

Winter Semester 2022/2023

Version: 11. Januar 2023